298a Design Document

UART Transmitter + Receiver

Peter and John Zhang

Introduction

This project implements a self-contained UART transmitter and receiver with built-in Hamming(7,4) error-correction, synthesized in Verilog for the Tiny Tapeout flow. By inserting a lightweight Hamming encoder before framing each byte and a matching decoder after reception, it guarantees single-bit error detection and correction over an asynchronous serial or wireless link. Designed for the ECE 298A's prototyping lab, this design demonstrates how digital communication, finite-state controllers, and error correcting codes can be combined into a compact, silicon-ready IP block.

Objectives & Main Function

The main objective of this project is to implement functional components of a UART communication system in Verilog. The chip design is separated into 2 distinct sections:

- UART Transmitter
- UART Receiver

The transmitter and receiver both integrate a Hamming(7,4) encoder/decoder for single-bit error detection and correction—delivering a robust, error-corrected asynchronous serial link and demonstrating key digital-design skills (finite-state machines, clock-division, coding theory, and testbench verification) in the ECE 298A prototyping flow.

The main function of this project is to reliably send and receive 4-bit data over an asynchronous serial link by:

1. **Transmitter:** converting parallel bytes into start-framed Hamming-encoded bit streams
2. **Receiver:** sampling and reassembling those streams, correcting any single-bit errors, and outputting the original bytes

Together, this ensures robust, error-corrected communication between two Verilog IP blocks.

Design Specifications

Our design specs break down into 2 parts – **components** and **configuration parameters**:

[1] Key Components

8x Oversampler	Generates periodic "sample" ticks at a fixed rate of 8x the baud rate.
Finite State Machine	Controls the 4 phases: 1. IDLE – line should be **1 (hi)** 2. START – reads 8 cycles of **0 (lo)** 3. DATA – reads 8 bits (64 cycles of data) 4. STOP – reads 8 cycles of **1 (hi)**
Hamming(7,4) Encoder/Decoder	Inserted before transmitting and after receiving 4 bits of data. TX Encoder: - receives [d0, d1, d2, d3] as input data - calculates 3 check bits - formats and sends 7 bits of data from LSB to MSB - outputs [c0, c1, d0, c2, d1, d2, d3] RX Decoder: - receives [c0, c1, d0, c2, d1, d2, d3] bits of data - calculates 3 parity bits - performs bit correction - outputs [d0, d1, d2, d3] parallel bits of data
Serial I/O Streams	Input – The single-bit RX line on the receiver side Output –The single-bit TX line on the transmitter side

[2] Configuration Parameters

On both sides of the UART link, we have the following parameters:

Clock Speed	50MHz
Baud Rate	650KHz
UART Communication Bits	Communication Protocol bits: - **IDLE** bit = 1 - **START** bit = 0 - **STOP** bit = 1

IO Design

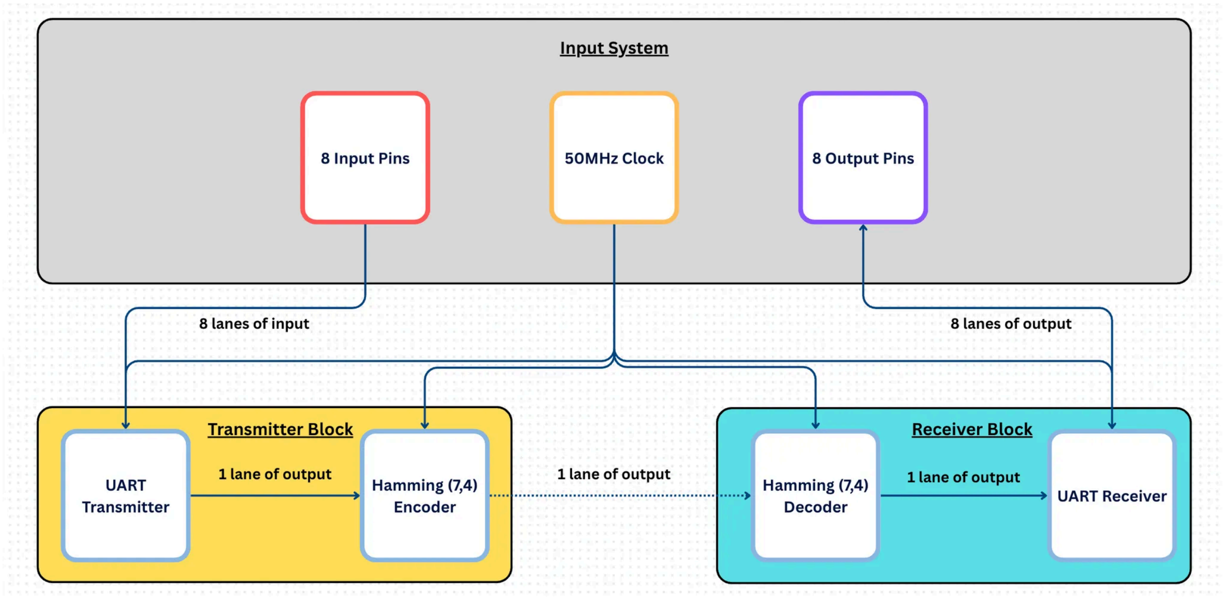
[1] IO Lines for General Usage

Proposed I/O Line	**Description**
tx_data_in[3:0] (4 lines)	Feeds parallel 4 bits of data into the TX
tx_data_out (1 line)	Receives 1 line of UART data stream from the TX
rx_data_in (1 line)	Feeds 1 line of UART data as a stream into the RX
rx_data_out[3:0] (4 lines)	Receives 4 lines of parallel bits from the RX
clk (1 line)	Shared system clock for all components
enable (1 line)	Actives/Deactivates system

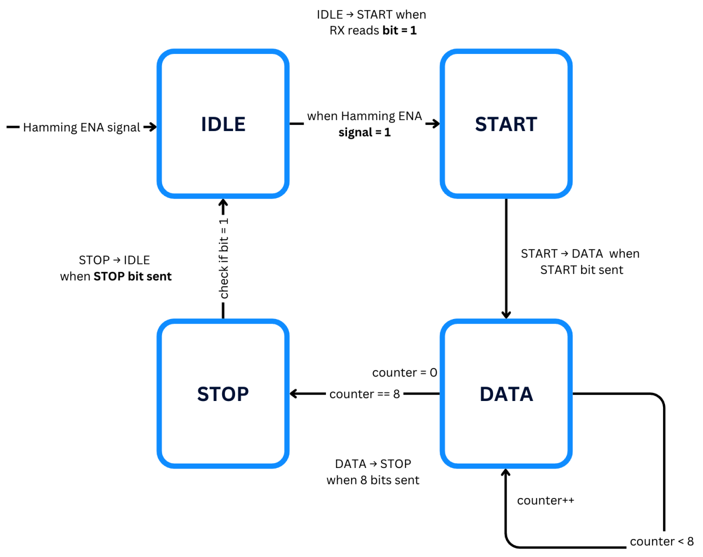
[2] Debug IO Lines

Debug I/O Line	**Description**
tx_check_bits[2:0] (3 lines)	Receives calculated check bits for TX
tx_state_bits[1:0] (2 lines)	Receives state bits for TX
rx_parity_bits[2:0] (3 lines)	Receives calculated parity bits for RX
rx_state_bits[1:0] (2 lines)	Receives state bits for RX

Block Diagram

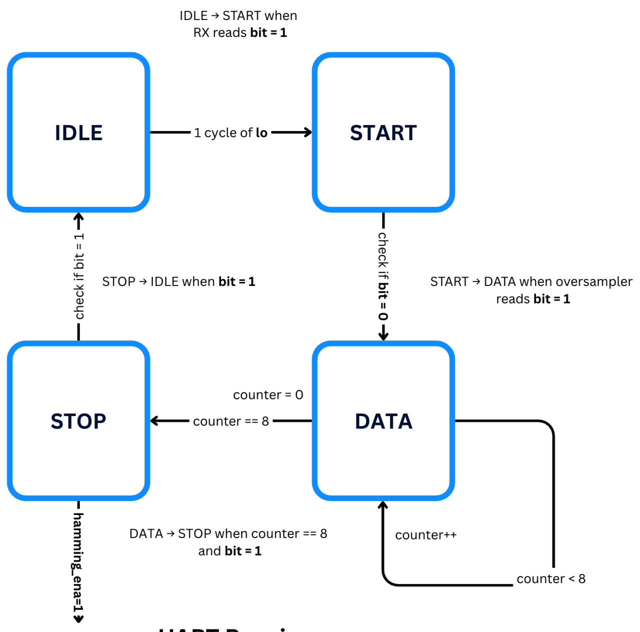


UART Transmitter State Machine


IDLE → START when RX reads bit = 1
Hamming ENA signal
IDLE
when Hamming ENA signal = 1
START
START → DATA when START bit sent
STOP → IDLE when STOP bit sent
check if bit = 1
STOP
counter = 0
counter == 8
DATA
DATA → STOP when 8 bits sent
counter++
counter < 8

UART Transmitter

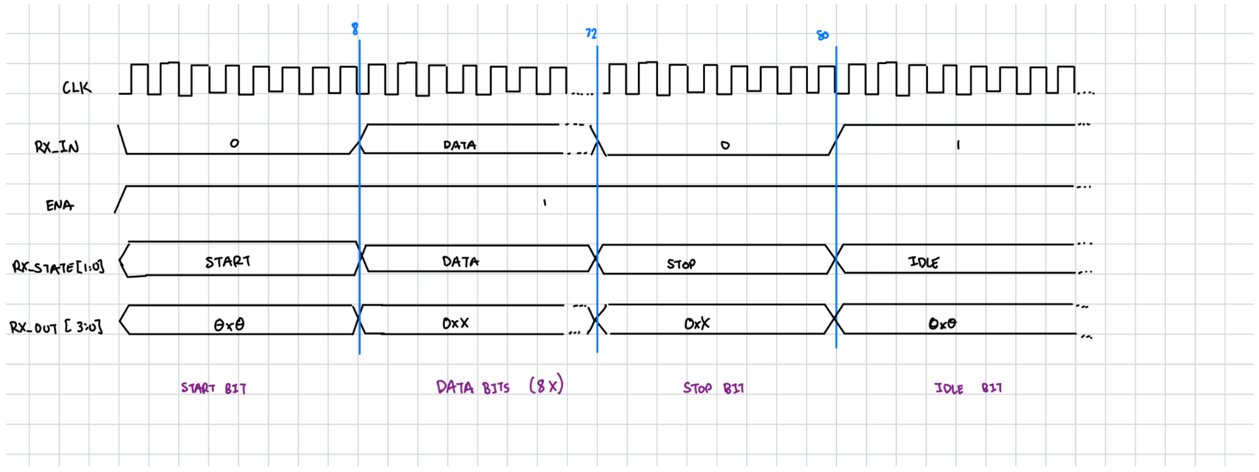
UART Receiver State Machine



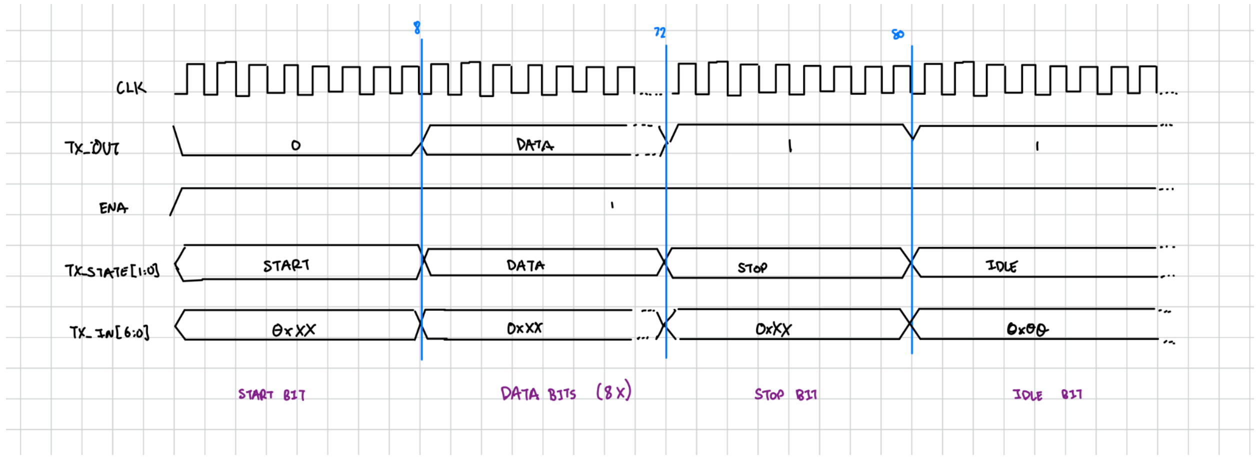
UART Receiver

Timing Diagrams

UART Receiver



UART Transmitter



Testing & Verification Plan

Here is a description of the tests in your test.py file:

test_full_hamming_code

This test verifies the UART transmitter and Hamming encoder by iterating through all possible 4-bit input values. For each input, it checks that the encoder produces the correct Hamming(7,4) codeword when there are no errors, and that the output changes appropriately when one or two bits are intentionally flipped to simulate errors. The test asserts that the codeword matches the expected value in the error-free case and differs from the expected value in the error cases.

test_error_free_data

This test checks the UART receiver and Hamming decoder by sending a valid, error-free Hamming(7,4) codeword over the UART interface. It waits for the decoder to process the input, then verifies that the decoded data matches the expected value, the syndrome output indicates no error, and the valid signal is asserted. The test logs the results and asserts that all outputs are correct for an error-free transmission.

test_single_bit_error

This test evaluates the UART receiver and Hamming decoder by sending a Hamming(7,4) codeword with a single bit error over the UART interface. After the decoder processes the input, the test checks that the syndrome output is non-zero (indicating an error was detected), the decoded data matches the expected value (showing the error was corrected), and the valid signal is asserted. The test logs the results and asserts that the decoder correctly detects and corrects the single-bit error.